13｜SVC：如何实现视频编码可伸缩？

2021-12-20 李江

《攻克视频技术》 课程介绍 >

讲述：李江
时长 17:21 大小 15.90M

你好，我是李江。

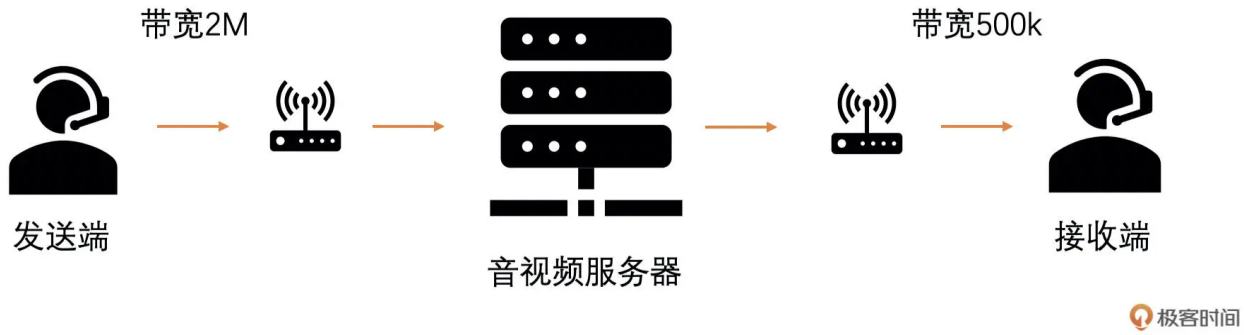
前面我们用了 4 节课的时间分别讲述了如何将视频编码码流打包成 H264，如何预测网络带宽，如何做好码控来控制视频发送的速率，如何分析视频的花屏和卡顿等问题。基本上循序渐进地将视频传输中最重要的一些知识点都讲解了一遍，并对里面几个重要的算法进行了深入的研究。

今天，我们再讲述一个视频会议场景中经常会使用的视频编码传输相关的技术——SVC 编码，也叫做可伸缩视频编码。它的作用是可以实现在一个码流里面包含多个可解码的子流，服务器可以根据接收端的网络状况，下发对应码率的码流，从而实现可伸缩性。

为什么需要 SVC

2020 年全球爆发新冠疫情，很多公司为了员工的安全，实行在家办公的政策。视频会议一时成为了工作中必不可少的日常工作活动。很多大型公司可能会出现一次几十、上百个人参加视频会议的情况。对于视频会议技术商来说，如何提供几十、上百个人的高质量视频通话技术是一个难题。为什么呢？

比如说，我和你两个人进行视频通话，我是发送端，网络非常好，你是接收端，网络比较差。发送端和接收端之间的视频通路如下图所示：



在带宽预测这节课里面我们讲过，由于服务器到接收端的网络比较差，那么最后会引起：

一组视频 RTP 包的接收时长很长，而一组视频 RTP 包的发送时长比较小；

或者发送端的视频 RTP 包发送给接收端之后，网络中丢包率很高。

如果不做带宽预测和码控的话最终接收端看到发送端的画面会非常卡。

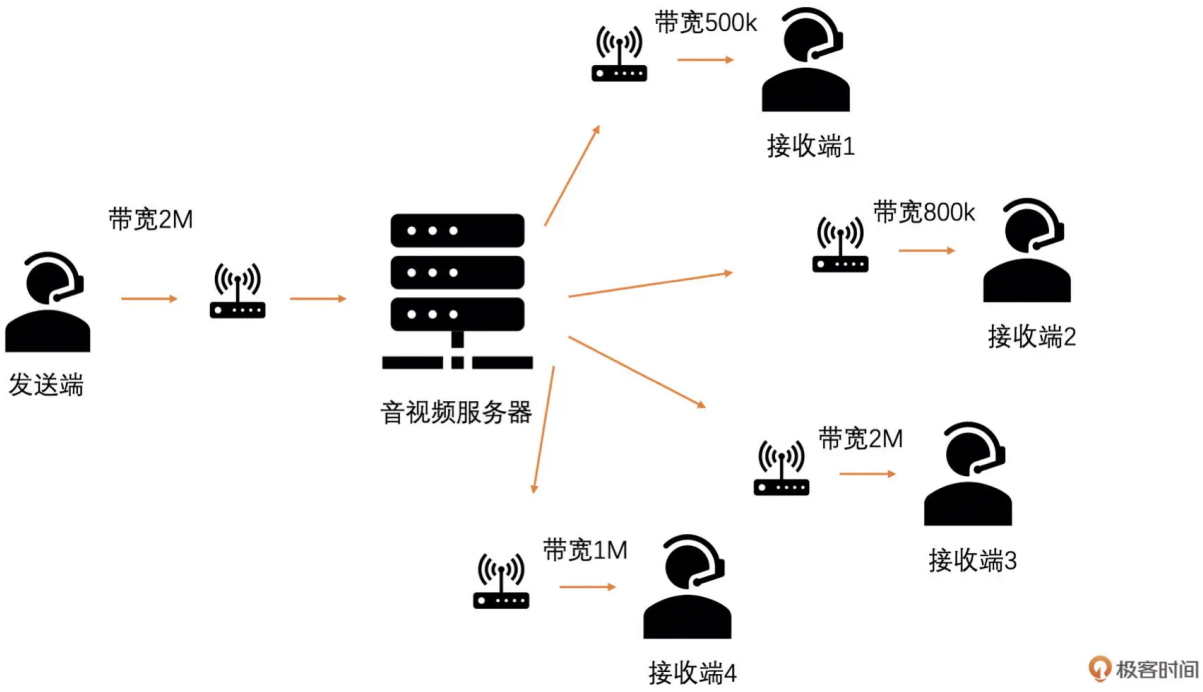
当然，我们肯定是会做带宽预测和码控的。遇到这种情况，发送端通过基于延时和基于丢包的带宽预测算法估算出发送端到接收端之间的网络带宽值。得到这个带宽值之后，参考第 11 讲，发送端的视频码控算法就会将码率降下来，同时，码率下降引起 QP 上升，画面质量下降，但是流畅性变好，不会一直卡死。

这是 1 对 1 视频会议场景遇到网络不好时的拥塞控制策略。

想象一下，现在有 10 个人参加视频会议，我是主持人，也是视频发送端。我们简化一下场景，我的画面需要发给其他的 9 个观众，而观众的画面不会发送给其他人。10 个人都在自己的家里面办公，每个人的网络状况都不一样。有的人网络非常好，带宽有 100M，而有

的人网络非常差，只有 500～600k，而且还因为使用无线网络丢包率很高。而我的网络比较中等，2M 带宽，有线网络丢包率非常小。

那现在开始视频会议，带宽预测算法开始工作，预测出我和其他 9 个人之间的链路带宽有 2M、1M、500k、800k 等很多个不同的带宽大小。假设忽略音频和其它数据占用的码率，所有的带宽都设置给视频的话，那请问现在设置给视频码控算法的目标码率应该是多少？



设置为 2M？那么带宽只有 500k、800k 的人看到的画面就是一直卡死的。那我们选择最小值 500k 不就可以了吗？是的，选择最小值 500k，那么所有人都可以流畅地看到画面。可是画面非常糊，质量很差。

想象一下，如果 100 人参加视频会议，99 个人的网络带宽是 100M，一个人带宽是 500k，最后我们选择 500k 的视频码率，合理吗？其他 99 个人为这一个人的网络不好一直观看质量非常差的画面。这太不公平了。有什么办法能很好地解决这个问题呢？这就是今天的主角——SVC。

什么是 SVC

SVC 是指一个码流当中，我们可以分成好几层。比如说分成三层：

第 0 层是最底层，可以独立进行编解码，不依赖第 1 层和第 2 层；

第 1 层编解码依赖于第 0 层，但是不依赖于第 2 层；

第 2 层的编解码需要依赖于第 0 层和第 1 层；

并且，**第 0 层质量最低，第 0 层加第 1 层次之，三层加在一起的时候质量最高。**注意这里的质量不是直接指的画面质量，而是帧率、分辨率的高低所代表的质量。

这样分层有什么好处呢？好处就是我们编码一个码流，可以组合出好几个不同的可解码码流出来。比如说上面三层 SVC 的例子：第 0 层就是一个可以独立解码的码流；第 0 层加上第 1 层也是一个可以独立解码的码流；第 0 层加上第 1 层和第 2 层也是一个可以解码的码流。

对于网络差的人，服务器给他转发第 0 层码流对应的 RTP 包；对于网络中等的人来说，服务器给他转发第 0 层加第 1 层码流对应的 RTP 包；对于网络很好的人，服务器给他直接转发所有层码流的 RTP 包。这样是不是就对大家都比较公平了。那具体怎么实现 SVC 分层编码呢？服务器又怎么转发呢？这里我给出我的思路，可供你参考。

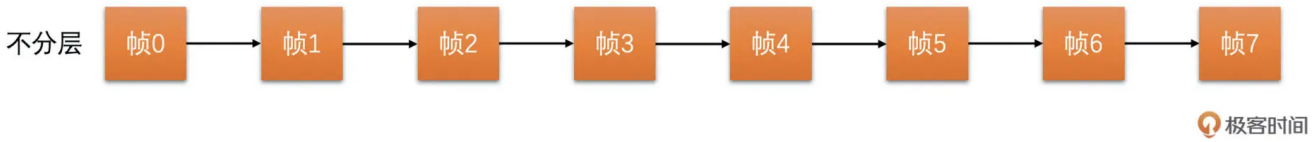
SVC 的分类

根据是在帧率上做 SVC 还是在分辨率上做 SVC，我们可以将 SVC 分为时域 SVC 和空域 SVC 两种。接下来我们逐一看一下。

时域 SVC

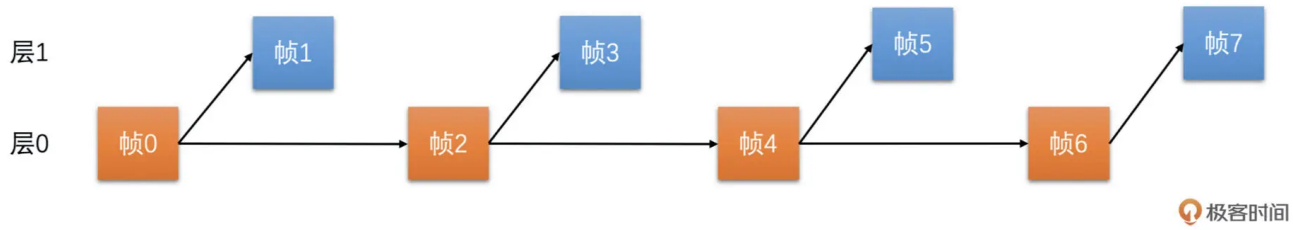
首先，第一种 SVC 分层编码方式是我们可以在帧率上做 SVC，这种 SVC 称之为时域 SVC。

帧率上做 SVC 是什么意思呢？我们在前面的课中讲过，一般我们在 RTC 场景中选择使用连续参考的参考结构来做编码。如下图所示：



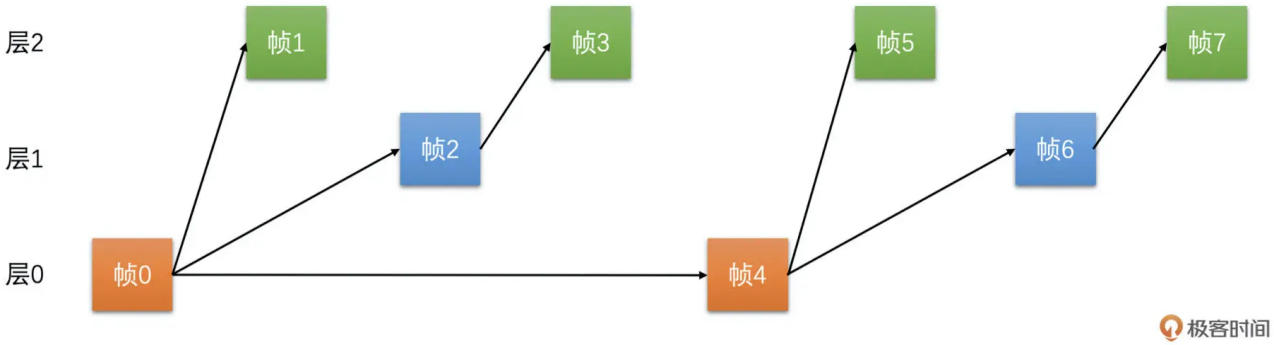
这种参考结构非常简单，但是有一个很大的问题就是只要有一帧被丢弃或不完整，就会导致后面的帧都不能解码，强行解码就会出现花屏（可以参考第 07 讲和第 12 讲）。

因此，如果是这种编码参考结构的话，就没有可伸缩性了。也就会产生前面多人视频会议的问题。我们把参考帧结构稍微换一下，隔一帧参考一帧，变成一个两层的结构，就可以解决连续参考的问题，如下图所示：



在图中，帧 0 是 I 帧不需要参考，且是第 0 层的帧。帧 1 是 P 帧，参考帧 0，且是第 1 层的帧。帧 2 是 P 帧，参考帧 0，不参考帧 1，是第 0 层的帧。帧 3 是 P 帧，参考帧 2，是第 1 层的帧。一直用这种模式不断地循环下去。

下面我们再来看一个三层时域 SVC 编码的参考帧结构，如下图所示：



在图中，帧 0 是 I 帧不需要参考，是第 0 层的帧。帧 1 是 P 帧，参考帧 0，与两层时域 SVC 不同，它是第 2 层的帧。帧 2 是 P 帧，参考帧 0，不参考帧 1，是第 1 层的帧。帧 3 是 P 帧，参考帧 1，是第 2 层的帧。帧 4 是 P 帧，参考帧 0，是第 0 层的帧，帧 5 是 P 帧，参考帧 4，是第 2 层的帧。不断按照这个模式循环下去。

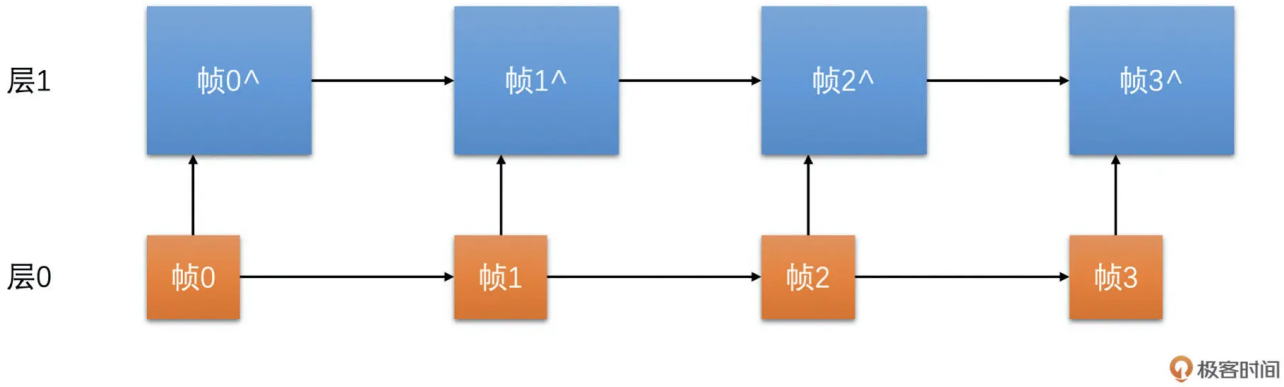
这个就是时域 SVC 编码。它的优点是什么呢？**它通过调整参考帧结构就能实现分层编码。低层的帧不会参考高层的帧。**如果我们丢弃高层的帧，低层的帧也是可以顺利地完成解码而不会出现花屏的，只是帧率会降低。但是相比连续参考结构中丢失一帧就直接卡死的体验要好很多。

同时，因为只需要调整一下参考结构，本身常用的编码标准都支持这种参考帧选择的方式，是符合常规标准的。因此，解码器都支持，没有兼容性问题。

但是它也有缺点。我们在第 07 讲中提到过，一般自然运动是连续的，选择前一帧作为参考帧一般压缩率会比较高，因为前后相邻的两帧很相似。**而时域 SVC 这种跨帧参考的方式会使得压缩率有一定的下降。**两层 SVC 编码效率大概下降 10%，三层大概下降 15%。

空域 SVC

下面，我们介绍另一种 SVC 编码，空域 SVC。**空域 SVC 是在分辨率上做分层。**比如说，我们现在需要编码一个 720P 的视频。我们分成两层：第 0 层是 360P 的分辨率；第 0 层加第 1 层是 720P 的分辨率。如下图所示：

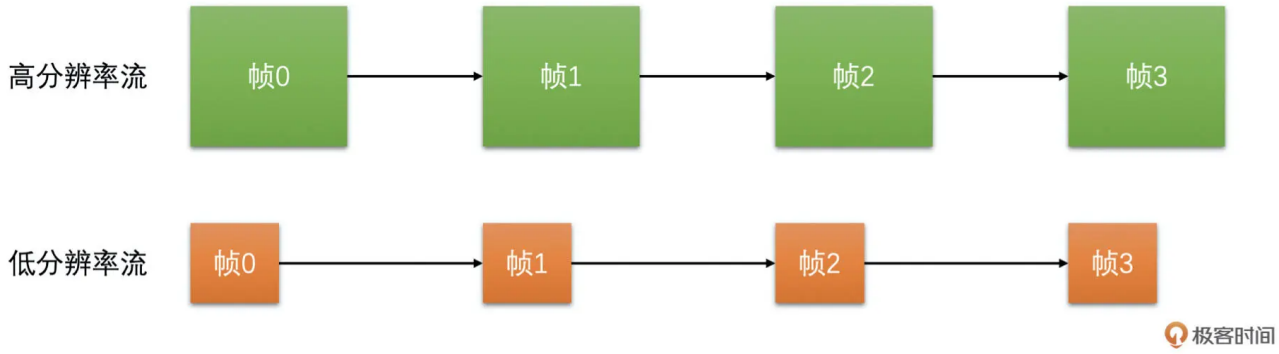


空域 SVC 的优点也是我们可以在一个码流当中分出多个码流出来。比如说，两层空域 SVC，第 0 层是一个可以独立解码的码流，只是分辨率是 360P。第 1 层依赖于第 0 层，两个层次加起来是 720P 分辨率的码流。每个不同的分辨率都对应不同的码率。因此，也可以用来解决多人视频会议的问题，只是丢弃了高层次的层之后分辨率会变小。

但是我必须要说明一下，**H264、H265、VP8 这些常用的编码标准（除了扩展）都是不支持空域 SVC 的。**因此，市面上的绝大多数的解码器也都不支持空域 SVC 这种一个码流里面含有多种分辨率的视频码流解码。所以现在很少会使用空域 SVC，也很少有编码器实现

空域 SVC。并且，这种多分辨率的空域 SVC 相比多个编码器编码不同分辨率的方式，在压缩率上也没有多少优势，而且还不符合常规的标准。

因此，在 WebRTC 中直接使用多个编码器编码多种分辨率的方式代替空域 SVC。



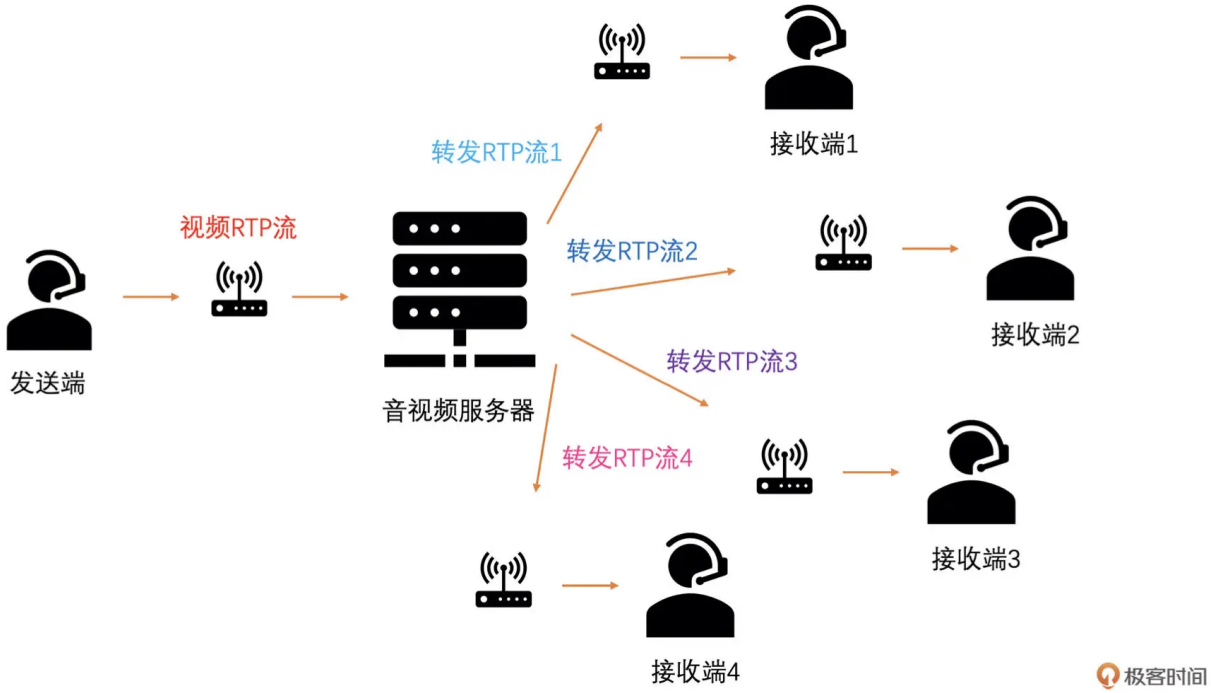
所以，我们接下来不会对空域 SVC 展开讨论。你可以当作是一个知识点了解一下就可以。

时域 SVC 如何实现可伸缩

下面我们再来看一下时域 SVC 如何做到给不同带宽的接收端转发不同帧率和码率的视频流。当然这个只是我的一些经验之谈。你可以参考一下。

首先，我们需要一些字段来描述码流中当前帧的层号、帧序号等 SVC 信息。因为这些字段只有在编码器编码的时候才知道。我们需要在编码出来一帧之后，在 RTP 包里面打包上这些信息发送给服务器和接收端。为什么需要告诉服务器和接收端呢？我们先来讲讲服务器如何根据网络情况做分层转发策略。

一般来说，视频会议使用如下的架构做视频数据转发。



服务器到接收端的链路上，服务器是发送端，在服务器上也需要做带宽预测，预测算法是一样的（可以参考第 10 讲）。

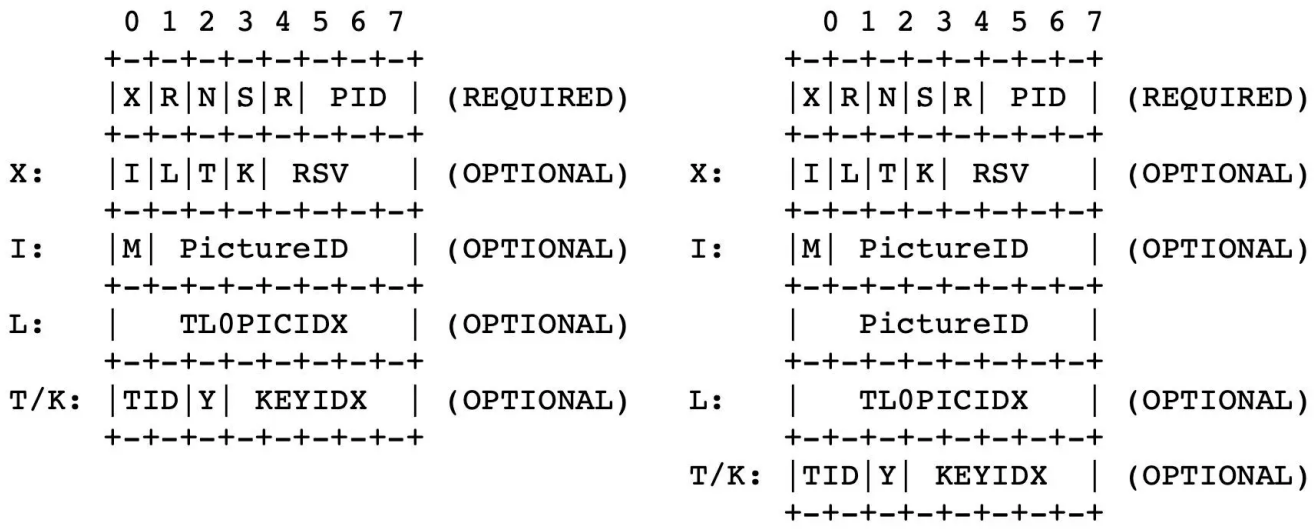
服务器会预测得到每一个接收端和服务器之间链路的带宽值。发送端发送 RTP 包到服务器，服务器需要通过计算 RTP 包的大小和当前 RTP 包所属的帧属于哪一层得到每一层对应的码率。这样服务器在转发的时候，就可以根据到接收端之间链路的带宽值和对应的每一层的码率来选择到底转发几层。

比如说，视频的码率是 2M，时域 SVC 编码，总共是 3 层，总帧率是 24fps。第一层帧率是 6fps，码率是 500k；第二层帧率是 6fps，码率是 500k；第三层帧率是 12fps，码率是 1M（这里假设码率按帧数平均分配）。

假设某一个接收端只有 600k，那服务器就只转发第一层给它，第二层第三层不转发。另一个接收端有 1.5M，那我们就转发第一层和第二层给它，而第三层不转发。还有一个接收端是 10M 的带宽，我们就转发一二三层给它。这就是时域 SVC 的服务器转发逻辑。

这个有一个重要的点就是，服务器如何知道每一个 RTP 包对应帧所在的层号，以及接收端如何知道当前帧可不可以解码，因为接收端是不知道服务器到底给自己转发几层的码流的。

这里我们可以参考 VP8 编码的 RTP 协议标准。**VP8 的 RTP 协议在 RTP 头和 VP8 码流数据的中间还有一个 RTP 描述头，这个描述头主要用来放帧号、层号等信息的。**具体如下图所示：



图片来源VP8的RTP文档

其中，几个重要的字段的解释如下：

I：占 1 位，表示有没有 PictureID 字段，为 1 表示有；

L：占 1 位，表示有没有 TL0PICIDX 字段，为 1 表示有；

T：占 1 位，表示有没有 Tid 和 Y 字段，为 1 表示有；

M：占 1 位，表示 PictureID 字段占 7 位还是 15 位，为 1 表示占 15 位；

PictureID：占 7 位或者 15 位，表示帧序号；

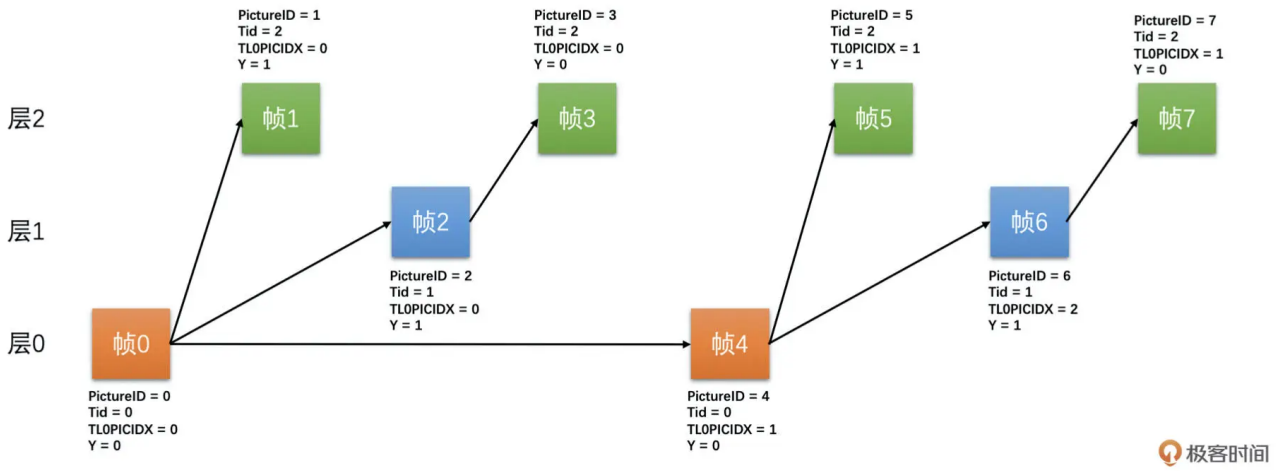
Tid：占 2 位，表示层号；

TL0PICIDX：占 8 位，表示当前帧所属的 SVC 单元，每过一个 Tid 为 0 的帧，TL0PICIDX 加 1；

Y：占 1 位，表示当前帧是不是只参考 Tid=0 的帧。

服务器可以从 RTP 描述头得到 RTP 包对应的层号。这样服务器就可以通过 RTP 的层号和 RTP 的包大小来估算每一层的码率了。

而接收端可以根据帧号、层号和层同步标志位等信息来判断当前帧是不是可以解码，而不用去解码视频码流。



从上图我们可以看到：

帧 0 是 IDR 帧，只要完整了就一定可以解码；

帧 1 是 P 帧，由于它的 Y 标志位为 1，代表它只参考了同一个 TL0PICIDX 中 Tid=0 的帧，也就是帧 0，因此，只要帧 0 可解，帧 1 就可以解码；

帧 2 判断逻辑同帧 1，只要帧 0 可解，帧 2 就可以解码，不依赖于帧 1 是不是可解；

帧 3 也是 P 帧，但是由于它的 Y=0，代表它不是只参考了 Tid=0 的帧，因此只有同一个 TL0PICIDX 中前面的帧都可解了才认为是可解的，也就是说只有帧 0、帧 1、帧 2 都可解它才可解，这里注意一下，因为帧 3 可以多参考，它可以同时参考帧 1 和帧 2，只是图中没有画出来；

帧 4 是 P 帧，但是它的 Tid=0，因此它只参考前面的帧 0，所以只要 TL0PICIDX-1 的 Tid=0 的帧可以解码，它就可以解码。也就是帧 0 可以解码，帧 4 就可以解码；

对于帧 5 判断同帧 1，帧 6 判断同帧 2，帧 7 判断同帧 3，一直循环下去；

我们可以看到帧 1、帧 3 丢弃了的话，并不影响帧 0 和帧 2 的可解码性判断。帧 1、帧 2、帧 3 都丢失了，也不会影响帧 4 的可解码性的判断。因此，**我们的服务器就可以通过丢层的方式来实现对不同带宽的接收端下发不同帧率码率的码流了。**

上面是 VP8 的时域 SVC 的 RTP 协议。那 H264 呢？H264 其实在标准的附录 G 直接定义了 SVC 的相关字段，也就是说在 H264 的编码码流里面就可以有 SVC 信息。如下图所示：

G.7.3.1.1 NAL unit header SVC extension syntax

nal_unit_header_svc_extension( ) {	C	Descriptor
idr_flag	All	u(1)
priority_id	All	u(6)
no_inter_layer_pred_flag	All	u(1)
dependency_id	All	u(3)
quality_id	All	u(4)
temporal_id	All	u(3)
use_ref_base_pic_flag	All	u(1)
discardable_flag	All	u(1)
output_flag	All	u(1)
reserved_three_2bits	All	u(2)
}		

图片来源H264标准文档

但是由于是附录 G 的内容，实现这一部分的解码器很少。因此不推荐使用这种方式传递 SVC 的相关信息。因为这种码流结构很多常规的 H264 解码器是不支持解码的，通用性不好，所以**我们建议使用 RTP 扩展头来传输时域 SVC 的信息。**

我们可以直接使用 VP8 的 RTP 描述头的格式，且编码码流还是保持常规标准的码流就可以，这样常规的 H264 解码器都能解码。服务器和接收端直接从 RTP 扩展头里面读取相关的 SVC 信息就可以了。而对于 SVC 编码，openh264 已经实现了最大 4 层的时域 SVC。你可以直接使用 openh264 就可以实现 SVC 编码了。

小结

总结一下，今天我们通过多人视频会议如何设置编码码率的问题引出了为什么需要使用 SVC 编码。SVC 编码可以在一个码流当中包含多个可以解码的子码流，这样服务器就可以根据接收端的带宽转发合适码率的子码流给接收端，从而达到可伸缩性。

并且，我们还介绍了两种类型的 SVC，主要包括时域 SVC 和空域 SVC。在之后，我们对服务器如何做时域 SVC 码流的转发做了详细的介绍。同时，我们还讨论了如何在 RTP 协议里面携带 SVC 信息，用于服务器做转发逻辑和接收端做解码性判断使用。

我们知道服务器会预测得到每一个接收端和服务器之间链路的带宽值，并通过计算 RTP 包的大小和当前 RTP 包携带的层号得到每一层对应的码率。然后，服务器再根据到接收端之间链路的带宽值和对应的每一层的码率来选择到底转发几层。

最后，接收端再根据 RTP 包携带的 SVC 信息来判断帧组完整之后可不可以解码，可以解码才能送解码器，不然就不能送去解码，防止出现花屏。这样我们就实现了可伸缩编码。

思考题

通过前面的学习，你知道哪些弱网对抗手段？

欢迎你在留言区和我分享你的思考和疑惑，你也可以把今天所学分享给身边的朋友，邀请他加入探讨，共同进步。下节课再见。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 1 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 12 | Jitter Buffer：拿什么拯救你，花屏卡顿问题

下一篇 14 | MP4 & FLV：不要再说AVI了

精选留言 (4)

写留言

paradise

2021-12-24

江哥，问一下，就是svc 编码采用rtp 包封装的形式，实际编码器出来的是普通码流，那这个码流是如何进行rtp svc 封装呢？ 是编码器提前设置好svc 的参考关系，然后从收到第一个I帧开始后就自动更新rtp 包里关于封装svc 编码的相关字段信息吗？

作者回复: 是的，webrtc里面的VP8编码是这样做的，外面固定好参考结构，编码的帧出来之后就按照固定的模式加上SVC信息。h264也可以这么做。或者编码器是你自己修改的，你也可以把信息从编码器里面带出来也可以，这样就可以不使用固定的参考结构了。

Geek_7de4c5
2021-12-21

如果是发送端网络不好，时域svc就失效了，有什么好的手段吗？

作者回复: 发送端网络不好直接通过发送端带宽预测得到带宽值再码控调节就可以了。

ripple
2021-12-20

simulcast是不是更常用些？

作者回复: 相比空域SVC，是的

晓龙
2021-12-20

SVC开始是否有前提，当接收端的方差达到某个阈值时在开启，或者发送端本身发送的帧率或者码率对网络要求很低，例如500k就可以，这时不启用SVC，可以更进一步增加压缩率。

作者回复: 1、如果接收端的带宽 > 对应层数的码流 + gap阈值，就发对应层数的码流。 我们需要设置这个gap阈值，这个值需要根据业务来自己设置。
2、一般来说我们整个视频会议的拥塞控制是带宽预测+码控+SVC，是不是需要启用SVC丢层策略关键还是在发送的码率和接收端的带宽大小对比来决定。我想你的意思是说我们可以把发送码率压缩到多个观众端的最小带宽。这个操作还是会有我之前说的问题，对于网络好的观众不公平。当然具体业务具体分析，如果你的业务这样做没什么问题的话也是可以的。

共 2 条评论 >